Ch. Lavanya Aishani* et al.
(IJITR) INTERNATIONAL JOURNAL OF INNOVATIVE TECHNOLOGY AND RESEARCH
Volume No.4, Issue No.5, August – September 2016, 3967–3973.

Efficient Usage of D Latch For Implementing A Reliable Low Power Area Carry Select Adder

CH. LAVANYA AISHANI
B.Tech Student.

SAILAJA SOLLA
PG Scholar, Dept of ECE
Pydah College of Engineering and Technology
Visakhapatnam, AP, India

YELLITHOTI SRAVAN KUMAR
Assistant Professor, Dept of ECE
Pydah College of Engineering and Technology
Visakhapatnam, AP, India

Abstract: **The Carry Select Adder is used in many systems to relieve the problem of carry propagation delay which is happen by independently generating multiple carries and to generate the sum then select a carry. Due to uses multiple pairs of Ripple Carry Adders (RCA) to generate partial sum and carry by considering carry input However, the CSLA is not time efficient, then by the multiplexers the final sum and carry are selected. The basic idea of this work is to achieve high speed and low power consumption by use Binary to Excess-1 Converter (BEC) instead of RCA in the regular CSLA. At the same time to further reduce the power consumption, a new approach of CSLA with D LATCH is proposed in this project.**

In the proposed scheme, before the calculation of-final-sum the carry select that is specified as CS operation is scheduled. For logic optimization of Carry selection bit patterns of two anticipating carry words that is corresponding to cin = 0 and 1 and fixed cin bits are used. Using optimized logic units an efficient CSLA design is obtained. The proposed Carry Select Adder design involves significantly less area and power than the recently proposed BEC-based CSLA.

Keywords: **CSLA; RCA; BEC; D-LATCH;**

I. INTRODUCTION

Now days the portability of the electronic component have rapid growth, the low power arithmetic circuit has become very important in VLSI industry. In The digital signal processor(DSP) main building block is the Multiplier-Accumulator (MAC) unit. Full Adder used as a part of the MAC unit uses fulladder as part which can significantly influences the efficiency of total system. Full Adder circuit is necessary for low power application due to the reduction in power consumption.The basic operation Carry Select Adder (CSLA) is parallel computation. CSLA generates many carriers and partial sum. Multiplexers selects The final sum and carry. In the CSLA architecture, Addition operation usually trembles widely the overall performance of digital systems and a crucial arithmetic function. The adders are most widely used in the electronic applications. In the year 2002, a new concept of adders are come into existence those are called as the hybrid adders and those are used for increeses the speed of addition process by Wang et al. the adders gives hybrid carry look-ahead/carry select adders design. In 2008, the new hybrid full adders are used for designing low power multipliers. In digital adders, the speed of addition is mainly based on the propagation delay ,it is having the limitation by propagating delay through adder. In VLSI Design one of the most important research is the area and power optimized data path logic systems. The adders are most widely used In electronic system and applications. If we want design multipliers the concept of adders comes in to the picture because the adders are part of the multipliers designs . As we know millions of instructions per second are performed in microprocessors. In the microprocessor device area and power consumption are the most important factors in the designing multipliers and adders. The power consumption and area should low in the microprocessors. Devices like computers Mobile phones, Laptops etc... those achieves more battery backup. So, a VLSI designer has to make perfect these parameters in a design. These are main constraints,so these are very difficult to achieve .so the constraints has to be made depending on demand or application of the circuit in the industry. the N full adders used in this architecture link together with N bit Ripple carry adder. In the ripple carry adder operation the carry out of previous full adder becomes the input carry for the next full adder. Like that sum and carry calculated at the end of the last full. As carry ripples from one full adder to the other, if the size of the full adder is high then it has a more delay.

II. OVERVIEW

The project consists of an efficient VLSI implementation of optimized power efficient CSLA using D-Latch. A novel and efficient VLSI

architecture is proposed and implemented for carry select adder.

The VLSI architecture has been authored in Verilog code for CSLA using D-Latch and its synthesis was done with Xilinx XST. Xilinx ISE Foundation 12.3has been used for performing mapping, placing and routing. For behavioral simulation place and route simulation ISE simulator has been used. The Synthesis tool was configured to optimize for area and high effort considerations. The interest of the project work is an attempt to obtain a CSLA using D-Latch architecture.

In our proposed design, we are implementing the CSLA in such a way that, it is balancing in between Area and Speed. Here we have reduced the Area while increasing the Speed. So, we can say that, it is the tradeoff between Area and Speed. In out design, we can achieve an Area efficient CSLA with optimized Speed.

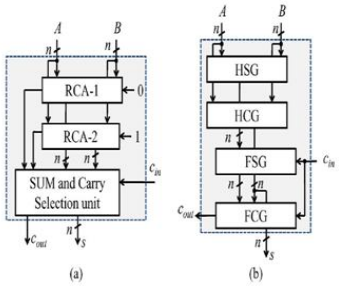


Fig. 1. (a) Conventional Carry Select Adder (b) The logic operations of the RCA is shown in split form.

As shown in Fig. 1(a), the SCG unit of the conventional Carry Select Adder is composed of two n-bit RCAs that is ripple carry adders , where HSG, HCG, FSG, and FCG represent half-sum generation, half-carry generation, full-sum generation, and full-carry generation, respectively. Where n is the adder bit-width. The logic operation of the n-bit RCA is performed in four stages: 1) half-sum generation which is specified as HSG 2) half-carry generation which is specified HCG 3) full-sum generation which is specified as FSG and 4) full carry generation which is specified as FCG. For example if two n-bit operands are added in the conventional CSLA, then RCA-1 and RCA-2 generate n-bit sum defined as s0 and s1 and output-carry which are as defined as $c_{out}^0$and c_{out}^1 are the corresponding to carry input as c_{in}=0 and c_{in}=1, respectively.

$$s_0^0(i) = A(i) \oplus B(i)\ c_0^0(i) = A(i) \cdot B(i) \quad (1a)$$

$$s_1^0(i) = s_0^0(i) \oplus c_1{}^0(i-1) \quad (1b)$$

$$c_1^0(i) = c_0{}^0(i) + s_0^0(i) \cdot c_1{}^0(i-1)\ c_{out}{}^0 = c1{}^0(n-1) \quad (1c)$$

$$s_0^1(i) = A(i) \oplus B(i)\ c_0^1(i) = A(i) \cdot B(i) \quad (2a)$$

$$s_1^1(i) = s_0^1(i) \oplus c_1{}^1(i-1) \quad (2b)$$

$$c_1^1(i) = c_0^1(i) + s_0^1(i) \cdot c_1{}^1(i-1)\ c_{out}{}^1 = c_1{}^1(n-1) \quad (2c)$$

As shown in Fig. 2, the RCA calculates n-bit sum $s_1^0$and c_{out}^0 corresponding to c_{in} = 0. The BEC unit receives s_1^0 and c_{out}^0 from the RCA and generates (n + 1)-bit excess-1 code. The most significant bit (MSB) of BEC represents c_{out}^1, in which n least significant bits (LSBs) represent s_1^1. The logic expressions

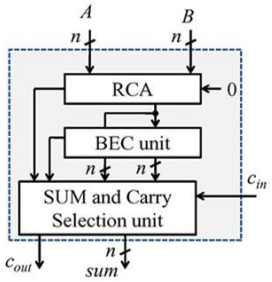


Fig. 2. Structure of the BEC-based CSLA; n is the input operand bit-width. of the RCA are the same as those given in (1a)–(1c). The logic expressions of the BEC unit of the n-bit BEC-based CSLA are given as

$$s_1^1(0) = s_1^0(0)\ c_1^1(0) = s_1^0(0) \quad (3a)$$

$$s_1^1(i) = s_1^0(i) \oplus c_1^1(i-1) \quad (3b)$$

$$c_1^1(i) = s_1^0(i) \cdot c_1^1(i-1) \quad (3c)$$

$$c_{out}^1 = c_1^0(n-1) \oplus c_1^1(n-1) \quad (3d)$$

for $1 \le i \le n-1$. We can find from (1a)–(1c) and (3a)–(3d) that, in the case of the BEC-based CSLA, c_1^1 depends on s_1^0, which otherwise has no dependence on $s1^0$ in the case of the conventional CSLA. The BEC method therefore increases data dependence in the CSLA.

We have considered logic expressions of the conventional CSLA and made a further study on the data dependence to find an optimized logic expression,

Binary Adder Notations and Operations:

From the verbal explanation of a half adder, we find that this circuit needs two binary inputs and two binary outputs. The input variables designate the augend and addend bits; the output variables produce the sum and carry. We assign symbols *x* and *y* to thetwo inputs and *S* (for sum) and *C* (for carry) to the outputs. The truth table for the half adder is listed in Table 4.3 . The *C* output is 1 only when both inputs are 1. The *S* output represents the least significant bit of the sum. The simplified Boolean functions for the two outputs can be obtained directly from the truth table. The simplified sum-of-products expressions are

$$Si = ai \wedge bi$$

$$Ci + 1 = ai \,.\, bi$$

The logic diagram of the half adder implemented in sum of products is shown in Fig. 4.5(a) . It can be

also implemented with an exclusive-OR and an AND gate as shownin Fig. 4.5(b) . This form is used to show that two half adders can be used to construct

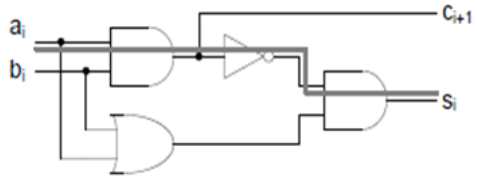


Fig.2.2: 1-bit Half Adder.

Consider a simple binary add with two n-bit inputs A;B and a one-bit carry-in cin along with n-bit output S.

S = A + B + Cin

Where A = an-1, an-2……a0; B = bn-1, bn-2……b0.

The + in the above equation is the regular add operation. However, in the binary world, only Boolean algebra works. For add related operations, AND, OR and Exclusive-OR (XOR) are required. In the following documentation, a dot between two variables (each with single bit), e.g. a _ b denotes 'a AND b'. Similarly, a + b denotes 'a OR b' and a _ b denotes 'a XOR b'. Considering the situation of adding two bits, the sum s and carry c can be expressed using Boolean operations mentioned above.

Si =ai xor bi

Ci + 1 =ai and bi

The Equation of Ci+1 can be implemented as shown in Fig.2.1. In the figure, there is a Half adder, which takes only 2 input bits.

The solid line highlights the critical path, which indicates the longest path from the input to the output. Equation of ci+1 can be extended to perform full add operation, where there is a carry input.

$$Si = ai \wedge bi \wedge ci$$

$$Ci+1 = ai \,\&\, bi + ai \,\&\, ci + bi \,\&\, ci$$

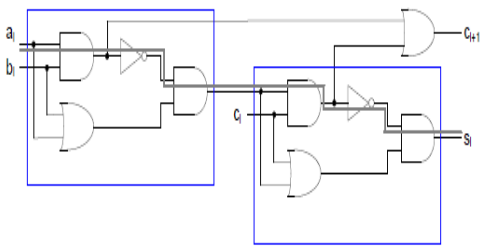


Fig.2.3: 1-bit Full Adder.

Addition of *n*-bit binary numbers requires the use of a full adder, and the process of additionproceeds on a bit-by-bit basis, right to left, beginning with the least significant bit. After the least significant bit, addition at each position adds not only the respective bits of the words, but must also consider a possible carry bit from addition at the previous position. A full adder is a combinational circuit that forms the arithmetic sum of three bits. It consists of three inputs and two outputs. Two of the input variables, denoted by *x* and *y* , represent the two significant bits to be added. The third input, *z* , represents the carry from the previous lower significant position. Two outputs are necessary because the arithmetic sum of three binary digits ranges in value from 0 to 3, and binary representation of 2 or 3 needs two bits.on for the CSLA.

Ripple carry adder:

A binary adder is a digital circuit that produces the arithmetic sum of two binary numbers. It can be constructed with full adders connected in cascade, with the output carry from each full adder connected to the input carry of the next full adder in the chain.

Addition of *n*-bit numbers requires a chain of *n* full adders or a chain of one-half adder and *n* 9 1 full adders. In the former case, the input carry to the least significant position is fixed at 0. Figure 2.4 shows the interconnection of four full-adder (FA) circuits to provide a four-bit binary ripple carry adder. The augend bits of *A* and the addend bits of *B* are designated by subscript numbers from right to left, with subscript 0 denoting the least significant bit. The carries are connected in a chain through the full adders. The input carry to the adder is *C*0, and it ripples through the full adders to the output carry *C*4. The *S* outputs generate the required sum bits. An *n* -bit adder requires *n* full adders, with each output carry connected to the input carry of the next higher order full adder.

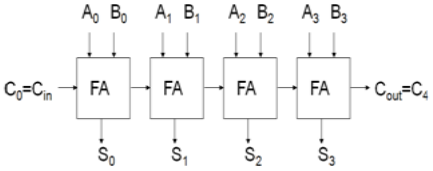


Fig.2.4:4-bRippleCarryAdder

Delay:

The latency of a 4-bit ripple carry adder can be derived by considering the worst-case signal propagation path. We can thus write the following expressions:

TRCA-4bit = TFA(A0,B0→Co)+T FA (C in→C1)+TFA (Cin→C2)+ TFA (Cin→S3)

And, it is easy to extend to k-bit RCA:

TRCA-4bit = TFA(A0,B0→Co)+(K-2)* TFA (Cin→Ci)+ TFA (Cin→Sk-1).

Carry Look-Ahead Adder:

Look ahead carry algorithm speed up the operation to perform addition, because in this algorithm carry for the next stages is calculated in advance based on input signals The carry propagation time is an important attribute of the adder because it limits the speed with which two numbers are added. Although the adder—or, for that matter, any combinational circuit—will always have some value at its output terminals, the outputs will not be correct unless the signals are given enough time to propagate through the gates connected from the inputs to the outputs. Since all other arithmetic operations are implemented by successive additions, the time consumed during the addition process is critical. An obvious solution for reducing the carry propagation delay time is to employ faster gates with reduced delays. However, physical circuits have a limit to their capability. Another solution is to increase the complexity of the equipment in such a way that the carry delay time is reduced.

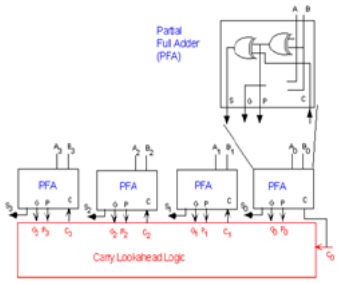


Fig.2.5: 4-bit Carry Look Ahead Adder

If 'X' and 'Y'are two inputs then if X=Y=1, a carry is generated independently of the carry from the previous bit position and if X=Y= 0, no carry is generated. Similarly if X ≠ Y, a carry is generated if and only if the previous bit-position generates a carry. 'C' is initial carry, "S" and "Cout" are output sum and carry respectively, then Boolean expression for calculating next carry and addition is:

Pi = Xi xor Yi --Carry Propagation

Gi = Xi and Yi -- Carry Generation

Ci + 1 =Gi or (Pi and Ci) -- Next Carry

Si = Xi xor Yi xorCi-- Sum Generation

Thus, for 4-bit adder, we can extend the carry, as shown below:

$$C1 = G0 + P0 \cdot C0$$

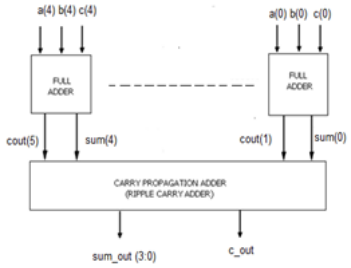
$$C2 = G1 + P1 \cdot C1 = G1 + P1 \cdot G0 + P1 \cdot P0 \cdot C0$$

$$C3 = G2 + P2 \cdot G1 + P2 \cdot P1 \cdot G0 + P2 \cdot P1 \cdot P0 \cdot C0$$

$$C4 = G3 + P3 \cdot G2 + P3 \cdot P2 \cdot G1 + P3 \cdot P2 \cdot P1 \cdot G0 + P3 \cdot P2 \cdot P1 \cdot P0 \cdot C0$$

Carry Save Adder:

The carry-save adder reduces the addition of 3 numbers to the addition of 2 numbers. The propagation delay is 3 gates regardless of the number of bits. The carry-save unit consists of n full adders, each of which computes a single sum and carries bit based solely on the corresponding bits of the three input numbers.



Let X and Y are two 4-bit numbers and produces partial sum and carry as S and C as shown in the below :

Si = Xi xor Yi ; Ci = Xi and Yi

The final addition is then computed as:

1. Shifting the carry sequence C left by one place.

2. Placing a 0 to the front (MSB) of the partial sum sequence S.

3. Finally, a ripple carry adder is used to add these two together and computing the resulting sum.

Carry Save Adder Computations :

X: 10011
Y: 11001
Z: + 01011
S: 00001
C: + 11011
SUM: 110111

Carry Select Adder:

A carry-select adder is divided into sectors, each of which – except for the least-significant –performs two additions in parallel, one assuming a carry-in of zero, the other a carry-in of one. A four bit carry select adder generally consists of two ripple carry adders and a multiplexer.

The carry-select adder is simple but rather fast, having a gate level depth of O(√n) . Adding two n-bit numbers with a carry select adder is done with two adders (two ripple carry adders) in order to perform the calculation twice, one time with the assumption of the carry being zero and the other assuming one.

III. THE D-TYPE FLIP FLOP

The working of D flip flop is similar to the D latch except that the output of D Flip Flop takes the state of the D input at the moment of a positive edge at the clock pin (or negative edge if the clock input is active low) and delays it by one clock cycle. That's why, it is commonly known as a delay flip-flop. The D Flip Flop can be interpreted as a delay line or zero order hold. The advantage of the D flip-flop over the D-type "transparent latch" is that the signal on the D input pin is captured the moment the flip-flop is clocked, and subsequent changes on the D input will be ignored until the next clock event.

D FLIP FLOPS AS DATA LATCHES:

As well as frequency division, another useful application of the D flip flop is as a **Data Latch**. A data latch can be used as a device to hold or remember the data present on its data input, thereby acting a bit like a single bit memory device and IC's such as the TTL 74LS74 or the CMOS 4042 are available in Quad format exactly for this purpose. By connecting together four, ***1-bit*** data latches so that all their clock inputs are connected together and are "clocked" at the same time, a simple "4-bit" Data latch can be made as shown below.

4-BIT DATA LATCH:

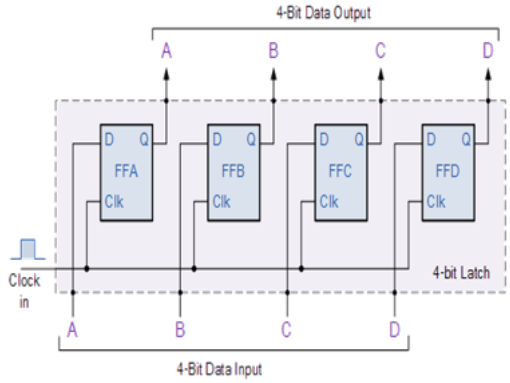


Fig. 3.3: 4-bit Data Latch

Proposed Adder Design:

The proposed carry select adder is based on the logic formulation given in (4a)–(4g), and the design of circuit is shown in Fig. 3(a). It is having one full sum generation (FSG) unit, one half sum generation (HSG) unit, one Carry Generation unit, and one Carry Sum unit. The carry generation unit is composed of two CGs (CG_0 and CG_1) corresponding to input-carry '0' and '1'. The half sum generation takes two n-bit operands to generate half-sum word s0 of width n-bit and half-carry word c0 of width n bits. Both CG_0 and CG_1 receive s_0 and c0 from the HSG unit and generate two n-bit full-carry words c_1^0 and c_1^1 corresponding to input-carry '0' and '1', respectively .The logic diagram of the HSG unit is shown in Fig. 3(b). The logic circuits of CG_0 and CG_1 are optimized to take advantage of the fixed input-carry bits. The optimized designs of CG_0 and CG_1 are shown in Fig. 3(c) and (d), respectively.

The CS unit selects one final carry word from the two carry words available at its input line using the control signal c_{in}. It selects c0 1 when c_{in} = 0; otherwise, it selects c_1^1. The CS unit can be implemented using an n-bit 2-to-l MUX. However, we find from the truth table of the CS unit that carry words c_1^0 and c_1^1 follow a specific bit pattern. If $c_1^0(i)$ = '1', then $c_1^1(i)$=1, irrespective of $s_0(i)$ and $c_0(i)$, for $0 \le i \le n-1$. This feature is used for logic optimization of the CS unit. The optimized design of the CS unit is shown in Fig. 3(e), which is composed of n AND–OR gates. The final carry word c is obtained from the CS unit. The MSB of c is sent to output as c_{out}, and (n − 1) LSBs are XORed with (n − 1) most significant bit(MSB) of half-sum (s_0) in the full sum generation(FSG)[shown in Fig. 3(f)] to obtain (n − 1) MSBs of final-sum (s). The least significant bit(LSB) of s0 is XOR with c_{in} to obtain the LSB of s.

The major speed limitation in any adder is in the production of carries and many authors have considered the addition problem. This logic can be implemented with Carry Select Adder usingD-Latch to achievepower-efficient and high-speed data path logic systems. The proposed 64-bit Carry Select Adder compared with the Carry Select Adder (CSLA) and Regular 64-bit Carry Select Adder and also with BEC 64-bit Carry Select Adder.

How the goal of fast addition is achieved using BEC together with a multiplexer (mux) is described in Fig.1.2, one input of the 8:4 mux gets as it input (B3, B2, B1, and B0) and another input of the Mux is the BEC output. This produces the two possible partial product results in parallel and the Muxes are used to select either BEC output or the direct inputs according to the control signal Cin,

The Boolean expressions of 4-bit BEC are listed below, (Note: functional symbols, ~ NOT, & AND, ^ XOR)

$$X0 = \sim B0$$

$$X1 = B0 \wedge B1$$

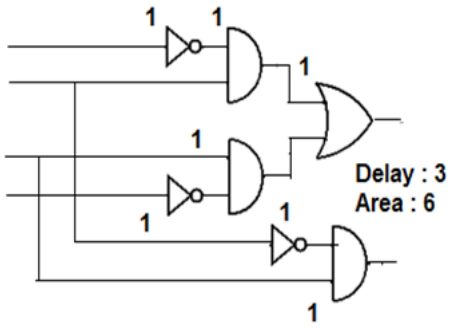
$$X2 = B2 \wedge (B0 \,\&\, B1)$$

$$X3 = B3 \wedge (B0 \,\&\, B1 \,\&\, B2)$$

The AND, OR, and Inverter (AOI) implementation of an2:1 MUX, FA are shown in below. The gates between the dotted lines are performing the operations in parallel and the numeric representation of each gate indicates how much delay is provided. AND, OR, and Inverter are the

basic gates for designing any digital circuits so the these are consider for the delay and area , those are having equal delay and area of 1-unit. If we add more number of gates through longest path it gives maximum delay.

The counting of AOI Gates a required for each logic block to decide the area evalution. Based on this approach, the CSLA adder blocks of 2:1 mux, Half Adder (HA), and FA are evaluated and listed in Table2.



IV. RESULTS

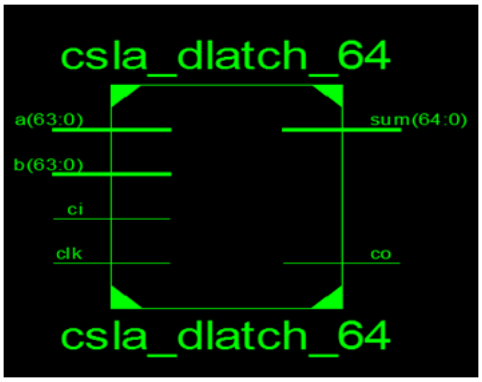


Fig 6: RTL Schematic view of 64 bit CSLA with D-Latch

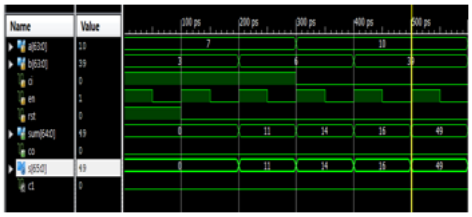
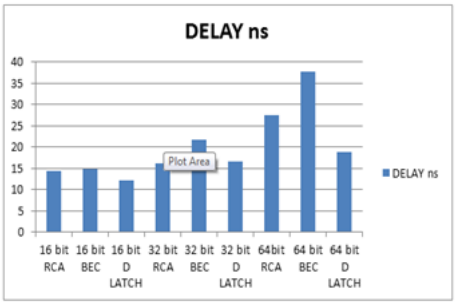


Fig 7: Waveform of 64 bit CSLA with D-Latch

	16 bit RCA	16 bit BEC	16 bit D LATCH	32 bit RCA	32 bit BEC	32 bit D LATCH	64bit RCA	64 bit BEC	64 bit D LATCH
Number of 4 input LUTs used	43	48	32	102	101	134	204	201	285
Available LUTs	9312	9312	9312	9312	9312	9312	9312	9312	9312
DELAY ns	14.362	14.73	12.211	16.117	21.756	16.506	27.539	37.772	18.873
Total supply power mw	76	76	76	76	76	76	76	76	76
Power Utilized mw	0.351	0.392	0.039	0.832	0.823	1.092	1.663	1.638	2.3235

Table 1: Results For The Selected Device XC3S1600E-5FG320



Graphical representation of Delay in 64-bit CSLA using D-Latch

V. CONCLUSION

Addition is the most common and often used arithmetic operation on microprocessor, digital signal processor, especially digital computers. Also, it serves as a building block for synthesis all other arithmetic operations. Therefore, regarding the efficient implementation of an arithmetic logic unit, the adder structures become a very critical hardware unit.

A D-LATCH based CSLA architecture is proposed in this project to reduce the delay of CSLA architecture than the recently proposed BEC based CSLA architecture. The functionality verification of the design is carried out by using ISE Simulator and the synthesis is also carried out by the XILINX ISE 12.3i.The HDL used for obtaining an RTL schematic and for designing the modules is VERILOG. From the graphs and the tables it is concluded that, the proposed D-LATCH based design is having less delay when compare to the BEC based and RCA based architectures.

VI. FUTURE SCOPE

With the increase in silicon densities, it is becoming feasible for compression systems to be implemented in a single chip. Here we have implemented CSLA using D-latch approach with less Delay. In future, we further reduce Area and Power parameters without the penalty of resources.

VII. REFERENCES

[1] B. Ramkumar and Harish M Kittur, "*Low Power and Area Efficient Carry Select Adder*" IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS-2011.

[2] Bedrij, O. J., (1962), "*Carry-select adder*," IRE Trans. Electron. Comput., pp.340–344 .

[3] Ceiang ,T. Y. and Hsiao,M. J. ,(*Oct. 1998),"Carry-select adder using single ripple carry adder*," Electron. Lett., vol. 34, no. 22, pp. 2101– 2103.

[4] Ramkumar,B. , Kittur, H.M. and Kannan ,P. M. ,(2010),"*ASIC implementation of*

modified faster carry save adder," Eur. J. Sci. Res., vol. 42, no. 1,pp.53–58,2010.

[5] J. M. Rabaey, Digtal Integrated Circuits—A Design Perspective. Upper Saddle River, NJ: Prentice-Hall, 2001.

[6] E. Abu-Shama and M. Bayoumi, "*A New cell for low power adders*," in Proc.Int. Midwest Symp. Circuits and Systems, 1995, pp. 1014–1017.

[7] Y. Kim and L.-S. Kim, "*64-bit carry-select adder with reduced area*,"Electron. Lett., vol. 37, no. 10, pp. 614–615, May 2001.

[8] Y. He, C. H. Chang, and J. Gu, "*An area efficient 64-bit square root carry-select adder for low power applications*," in Proc. IEEE Int. Symp. Circuits Syst., 2005, vol 4, pp.4082-4085.

